**Restoring division of two numbers:**

**Algorithm:**

Step 1: Start.

Step 2: Initialize and take number inputs in decimal.

Step 3: Initialize array of specific bit size Q, M, temp and Mcomp. Also initialize accumulator to 0, an integer a = 0 and counter = no. of bits.

Step 4: Convert the first decimal number to binary. If the number is negative, compute the two’s complement for the number and store it in Q.

Step 5: Convert the first decimal number to binary. If the number is negative, compute the two’s complement for the number and store it in Q.

Step 6: Find the 2’s complement of M and store it in Mcomp.

Step 7: Store content of Q in temp.

Step 8: if accumulator[0] = Q[0]; shift left accumulator and temp and add A and Mcomp.

Step 9: Initialize temp[n – 1] = 0.

Step 10: if accumulator[0] != Q[0]; add A and M and put temp[n – 1] = 1. Else temp[n – 1] = 0.

Step 11: count --.

Step 12: If count > 0, go to step 8.

Step 13: Stop.

**Source code:**

*#include<stdio.h>*

*#include <stdlib.h>*

*#include <math.h>*

*#define SIZE 5*

*void decimalToBinary(int n, int \*binary, int size) {*

*if (n < 0) {*

*n = abs(n);*

*}*

*for (int i = size - 1; i >= 0; i--) {*

*binary[i] = n % 2;*

*n = n / 2;*

*}*

*}*

*void twosComplement(int \*binary, int size) {*

*int carry = 1;*

*for (int i = 0; i < size; i++) {*

*binary[i] = binary[i] == 0 ? 1 : 0;*

*}*

*for (int i = size - 1; i >= 0; i--) {*

*binary[i] = binary[i] + carry;*

*if (binary[i] == 2) {*

*binary[i] = 0;*

*carry = 1;*

*} else {*

*carry = 0;*

*}*

*}*

*}*

*void addTwoBinaries(int \*binary1, int \*binary2, int size) {*

*int carry = 0;*

*for (int i = size - 1; i >= 0; i--) {*

*binary1[i] = binary1[i] + binary2[i] + carry;*

*if (binary1[i] == 2) {*

*binary1[i] = 0;*

*carry = 1;*

*} else if (binary1[i] == 3) {*

*binary1[i] = 1;*

*carry = 1;*

*} else {*

*carry = 0;*

*}*

*}*

*}*

*void shiftLeft(int \*binary1, int \*binary2, int size)*

*{*

*for (int i = 0; i < size - 1; i++){*

*binary1[i] = binary1[i + 1];*

*}*

*binary1[size - 1] = binary2[0];*

*for (int i = 0; i < size - 1; i++){*

*binary2[i] = binary2[i + 1];*

*}*

*binary2[size - 1] = 0;*

*}*

*void printBinary(int \*binary, int size) {*

*for (int i = 0; i < size; i++) {*

*printf("%d", binary[i]);*

*}*

*}*

*void printRow(int count, int \*accumulator, int \*temp, const char\* operation) {*

*printBinary(accumulator, SIZE);*

*printf(" | ");*

*printBinary(temp, SIZE);*

*printf(" | %d | %s\n", count, operation);*

*}*

*int main(){*

*int a1, b1, Q[SIZE], M[SIZE], accumulator[SIZE] = {0}, acc[SIZE], temp[SIZE], count = SIZE, twoCompM[SIZE];*

*printf("Enter the first number: ");*

*scanf("%d", &a1);*

*printf("Enter the second number: ");*

*scanf("%d", &b1);*

*printf("\n");*

*decimalToBinary(a1, Q, SIZE);*

*if (a1 < 0) {*

*twosComplement(Q, SIZE);*

*}*

*printf("First number in binary: ");*

*printBinary(Q, SIZE);*

*printf("\n");*

*decimalToBinary(b1, M, SIZE);*

*if (b1 < 0) {*

*twosComplement(M, SIZE);*

*}*

*printf("Second number in binary: ");*

*printBinary(M, SIZE);*

*printf("\n");*

*for (int i = 0; i < SIZE; i++){*

*temp[i] = Q[i];*

*}*

*for (int i = 0; i < SIZE; i++){*

*twoCompM[i] = M[i];*

*}*

*twosComplement(twoCompM, SIZE);*

*printRow(count, accumulator, temp, "Initialization");*

*printf("\n");*

*while (count > 0){*

*if (accumulator[0] == M[0]){*

*shiftLeft(accumulator, temp, SIZE);*

*printRow(count, accumulator, temp, "Left Shift");*

*for (int i = 0; i < SIZE; i++){*

*acc[i] = accumulator[i];*

*}*

*addTwoBinaries(acc, twoCompM, SIZE);*

*printRow(count, acc, temp, "A <-- A - M");*

*count--;*

*if (accumulator[0] != acc[0]){*

*temp[SIZE - 1] = 0;*

*printRow(count, accumulator, temp, "A <-- A + M; restored and count --");*

*}*

*else{*

*for (int i = 0; i < SIZE; i++){*

*accumulator[i] = acc[i];*

*}*

*temp[SIZE - 1] = 1;*

*printRow(count, accumulator, temp, "count --");*

*}*

*}*

*}*

*printf("\n");*

*printf("\n\n");*

*printf("The Quotient for the input division is: ");*

*printBinary(temp, SIZE);*

*printf("\n");*

*printf("The Remainder for the input division is: ");*

*printBinary(accumulator, SIZE);*

*return 0;*

*}*

**Sample input/output:**

* Input of 15 and 4

**A black screen with white text

Description automatically generated**

* Sample output for above input

**A computer screen shot of a number

Description automatically generated**

* Input of 15 and -4

A screenshot of a computer

Description automatically generated

* Sample output for above input

A screenshot of a computer program

Description automatically generated